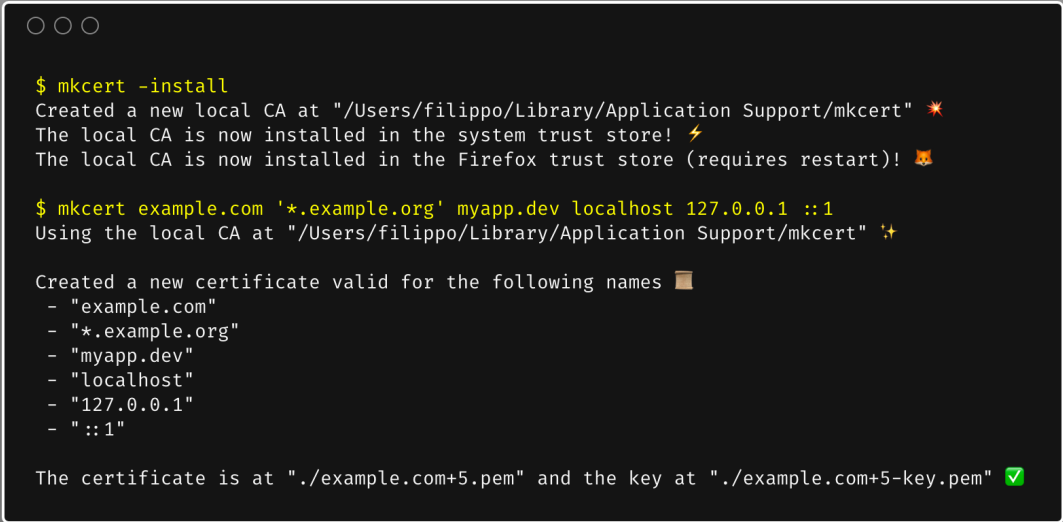


# FILIPPO.IO

Filippo Valsorda, 06 Jan 2019 on Go | TLS

## MKCERT: VALID HTTPS CERTIFICATES FOR LOCALHOST

*(or for any other name)*



```
$ mkcert -install
Created a new local CA at "/Users/filippo/Library/Application Support/mkcert" ✨
The local CA is now installed in the system trust store! ⚡
The local CA is now installed in the Firefox trust store (requires restart)! 🦊

$ mkcert example.com '*.example.org' myapp.dev localhost 127.0.0.1 ::1
Using the local CA at "/Users/filippo/Library/Application Support/mkcert" ✨

Created a new certificate valid for the following names 📋
- "example.com"
- "*.example.org"
- "myapp.dev"
- "localhost"
- "127.0.0.1"
- "::1"

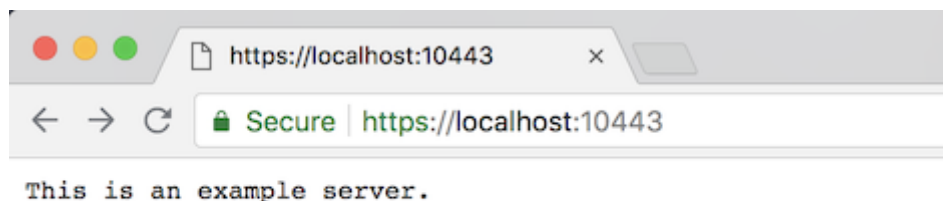
The certificate is at "./example.com+5.pem" and the key at "./example.com+5-key.pem" ✅
```

The web is moving to HTTPS, preventing network attackers from observing or injecting page contents. But

HTTPS needs TLS certificates, and while deployment is increasingly a solved issue thanks to the ACME protocol and Let's Encrypt, development still mostly ends up happening over HTTP because no one can get an universally valid certificate for *localhost*.

This is a problem because more and more browser features are being made available only to secure origins, and testing with HTTP hides any mixed content issues that can break a production HTTPS website. Developing with HTTPS should be as easy as deploying with HTTPS.

That's what **mkcert** is for.



mkcert is a simple by design tool that hides all the arcane knowledge required to generate valid TLS certificates. It works for any hostname or IP, including *localhost*, because it only works for you.

```
$ mkcert example.com example-staging.appspot.com localhost
Using the local CA at "/Users/filippo/Library/Application S
```

```
Created a new certificate valid for the following names 📄
- "example.com"
```

```
"example-staging.appspot.com"
```

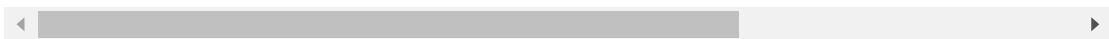
- "example-staging.appspot.com"
- "localhost"

The certificate is at `./example.com+2.pem` and the key at



Here's the twist: it doesn't generate self-signed certificates, but certificates signed by your own private CA, which your machine is automatically configured to trust when you run `mkcert -install`. So when your browser loads a certificate generated by your instance of mkcert, it will show up with a green lock!

```
$ mkcert -install
Using the local CA at "/Users/filippo/Library/Application S
The local CA is now installed in the system trust store! ⚡
The local CA is now installed in the Firefox trust store (r
```



It supports macOS, Linux, and Windows, and Firefox, Chrome and Java. It even works on mobile devices with a couple manual steps.

Also, unlike OpenSSL, it does the right thing by default, instead of forcing you to use a dozen flags and materialize a config file for each certificate. (That is, it uses Subject Alternative Names, instead of the 20-years-deprecated Common Name.)

The hardest part of the project, besides figuring out half a dozen different root stores, has been keeping the tool

simple and focused. There are adjacent use cases that mkcert might be good for, like acting as a CA infrastructure for microservices, but that's not what mkcert is for. mkcert is a development tool, and that focus allowed it to provide useful defaults and limit configuration options to virtually zero. Other tools can fill other gaps better.

One feature is left before mkcert is finished: an ACME server. If you are doing TLS certificates right in production, you are using Let's Encrypt via the ACME protocol. Development and staging should be as close to production as possible, so mkcert will soon act as an ACME server like Let's Encrypt, providing locally-trusted certificates with no verification. Then all you'll have to change between dev and prod will be the URL of the ACME endpoint.

As for now, mkcert is already stable, with 8 releases and almost 12k stars. You can install it from most package managers, or use the pre-built binaries. Please try it in your workflows, and report any usability issues. You might also want to follow me on Twitter.

**mkcert****https://localhost**

---

## Filippo Valsorda

Cryptographer on the Go team at Google. RC F'13, F2'17. You might know me as @FiloSottile.

Want to become a better programmer? [Join the Recurse Center!](#)

Sign up to my newsletter—[Cryptography Dispatches](#)—for more frequent, lightly edited writings on cryptography.

### Subscribe to Cryptography Dispatches:

Subscribe

Powered by [Buttondown](#).



All content copyright [Filippo.io](#) © 2020 • All rights reserved.

Proudly published with  **Ghost**